Real-Time Audio Translator Setup Guide

This guide helps you set up a real-time audio translator that captures system audio, transcribes it, and translates it - all running locally on your machine.

📋 Requirements

- Python 3.8 to 3.11 (3.12+ may have compatibility issues)
- Windows 10/11 with Stereo Mix or similar audio loopback
- 4GB+ RAM (8GB recommended for larger models)
- Audio playing through your system speakers

🚀 Quick Start

Step 0: Install Dependencies

```bash
python setup.py
```

This installs all required packages:

- numpy
- sounddevice
- faster-whisper
- transformers
- torch (optional but recommended)

Step 1: Find Your Audio Device

```bash
python step1_device_scanner.py
```

This script will:

- List all audio devices on your system
- Test each device to find system audio capture
- Create `audio_config.py` with optimal settings

Troubleshooting:

- Make sure audio is playing during the test
- If no devices work, enable "Stereo Mix" in Windows Sound settings
- Try different audio sources (YouTube, music player, etc.)

Step 2: Test Audio Capture

```bash
python step2_audio_capture_test.py
```

This verifies:

- Audio is being captured correctly
- Levels are appropriate
- Threshold is calibrated

You should see audio levels displayed in real-time.

Step 3: Test Transcription

```bash
python step3_whisper_integration.py
```

This tests real-time transcription:

- Choose model size (tiny = fastest, small = most accurate)
- Play videos or audio with speech
- See transcriptions appear in real-time

Step 4: Run Full Translator

```bash
python step4_complete_pipeline.py
```

The complete pipeline:

- Enter source language (or 'auto' for detection)
- Enter target language

- Choose model size
- Watch real-time translations!

🔧 Configuration

The `audio_config.py` file contains:

```python
DEVICE_ID = 21  # Your system audio device
SAMPLE_RATE = 48000  # Device sample rate
CHANNELS = 1  # Mono is more reliable
AUDIO_GAIN = 0.001  # Prevents overflow
ENERGY_THRESHOLD = 0.005  # Voice detection threshold
```

Adjust these if needed:

- **AUDIO_GAIN**: Lower if audio clips, higher if too quiet
- **ENERGY_THRESHOLD**: Lower to detect quieter speech

🌐 Supported Languages

Common language codes:

- **en** - English
- **es** - Spanish
- **fr** - French
- **de** - German
- **it** - Italian
- **pt** - Portuguese
- **ru** - Russian
- **zh** - Chinese
- **ja** - Japanese
- **ar** - Arabic
- **hi** - Hindi

📊 Performance Tips

1. **Model Selection**:
   - `tiny`: Fastest, ~1GB RAM, good for real-time

    - `base`: Better accuracy, ~1.5GB RAM
    - `small`: Best accuracy, ~2.5GB RAM

2. **Reduce Latency**:
    - Use `tiny` model
    - Install CUDA for GPU acceleration
    - Close other heavy applications

3. **Improve Accuracy**:
    - Use `base` or `small` model
    - Ensure clear audio source
    - Adjust threshold for your environment

❗ Common Issues

"No audio detected"

- Check Windows Sound settings
- Enable "Stereo Mix" or "What U Hear"
- Make sure audio is playing through speakers (not headphones only)

High latency

- Switch to `tiny` model
- Check CPU usage
- Consider GPU acceleration with CUDA

Poor transcription

- Try `base` or `small` model
- Check audio quality
- Ensure speech is clear in source

Installation errors

- Use Python 3.8-3.11 (not 3.12+)
- Update pip: `python -m pip install --upgrade pip`
- Install Visual C++ Build Tools for Windows

🔄 Workflow Summary

1. **Setup Phase** (one time):
   - Run `setup.py` to install dependencies
   - Run `step1_device_scanner.py` to configure audio
   - Verify with `step2_audio_capture_test.py`
2. **Usage Phase**:
   - Run `step4_complete_pipeline.py` for translations
   - Or `step3_whisper_integration.py` for transcription only
3. **Different PC**:
   - Copy all scripts to new PC
   - Run from Step 0 (setup.py)
   - Each PC needs its own device configuration

📁 File Structure

```
your_project/
│
├── setup.py                  # Install dependencies
├── step1_device_scanner.py     # Find audio device
├── step2_audio_capture_test.py # Test audio capture
├── step3_whisper_integration.py # Test transcription
├── step4_complete_pipeline.py  # Full translator
├── audio_config.py           # Auto-generated config
└── README.md                  # This file
```

🎯 Next Steps

After basic setup works:

1. Add GUI with tkinter
2. Add more language pairs
3. Optimize for your specific use case
4. Add text output/logging features

Good luck with your CS50 final project! 🎉